Отчёт по лабораторной работе №10

Борунов Семён

Содержание

# 1 Цель работы

Освоить работу с подпрограммами и отладчиком gdb.

# 2 Выполнение лабораторной работы

Создадим рабочую директорию и файл. Запишем туда программу из листинга, исправив опечатки. (рис. 1)

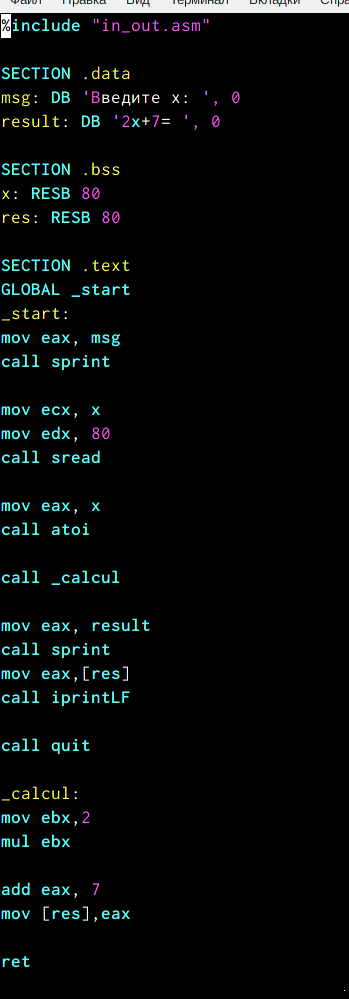


Рис. 1: -

напишем программу, имитирующую сложную функцию. Функции назовем \_calul и *subcalcul*(рис. 2)

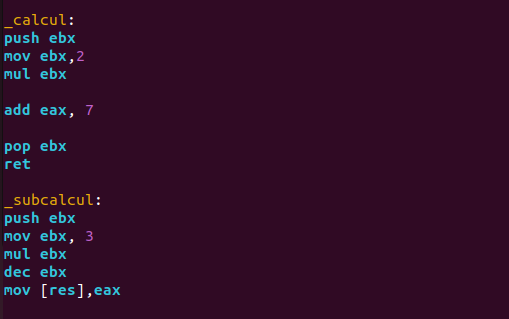


Рис. 2: f(g(x))

Проверим ее работу (рис. [-fig. 3)

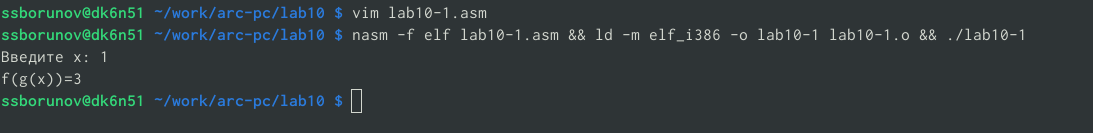


Рис. 3: -

Создадим файл lab10-2.asm и посмотрим, как она работает. Так же проассемблируем его с другими ключами, чтобы была возможность открыть этот файл через gdb. (рис. 4)

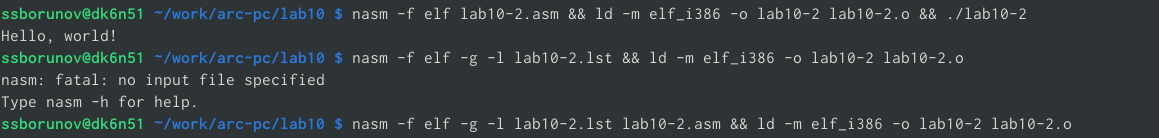


Рис. 4: -

Откроем lab10-2 с помощью gdb. Запустим ее там(рис. 5)

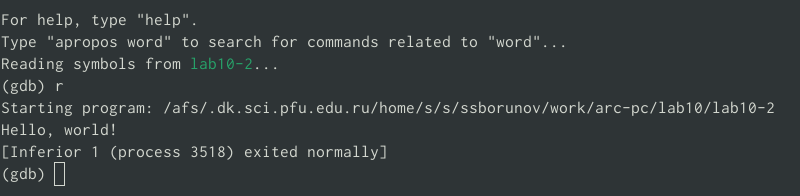


Рис. 5: -

Поставим точку останова(breakpoint) на метке \_start. Посмотрим дизассемеблированный код, начиная с этой метки. (рис. 6)

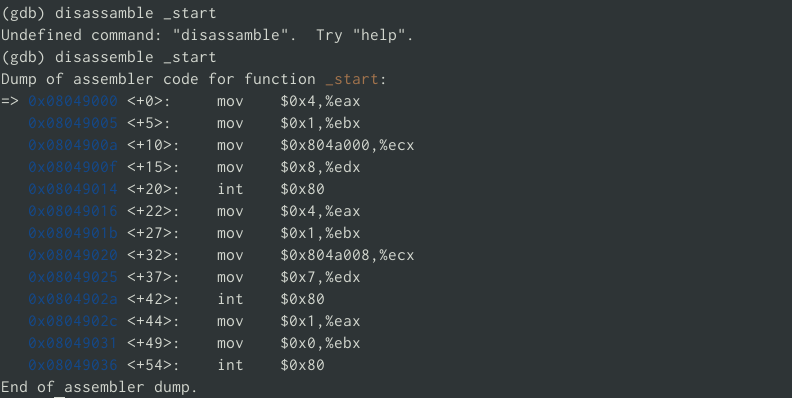


Рис. 6: -

Так же посмотрим как выглядит дизассемблированный код c синтаксисом Intel (рис. 7)

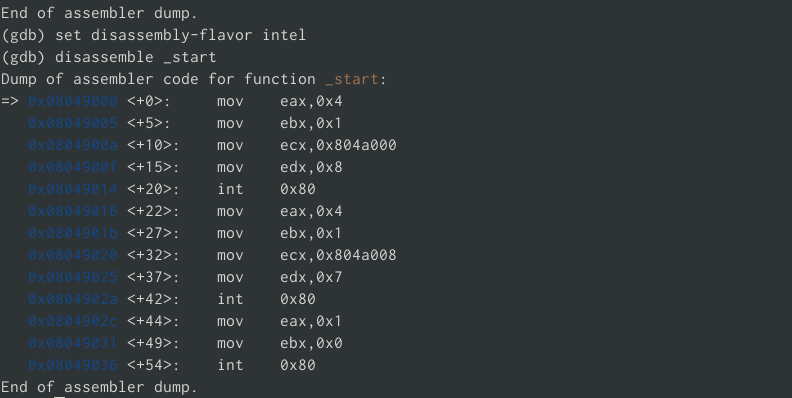


Рис. 7: -

В представлении ATT в виде 16-ричного числа записаны первые аргументы всех комманд, а в представлении intel так записываются адреса вторых аргумантов.

включим режим псевдографики, с помощью которго отбражается код программы и содержимое регистров(рис. 8)

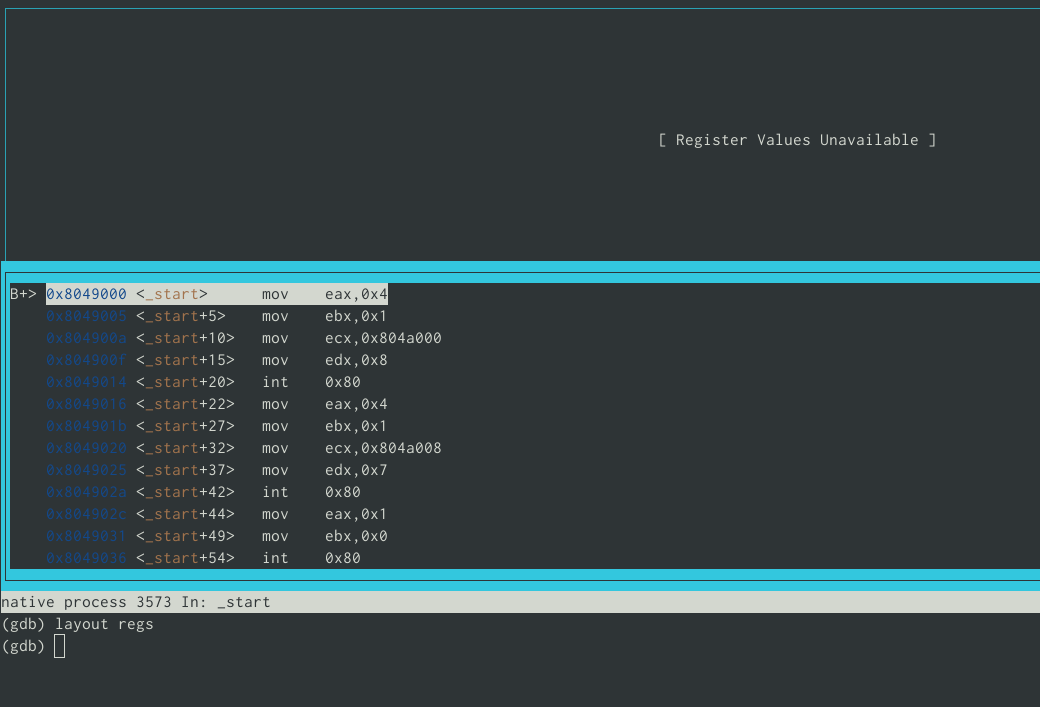


Рис. 8: -

Посмотрим информацию о наших точках останова. Сделать это можно коротко командой i b (рис. 9)

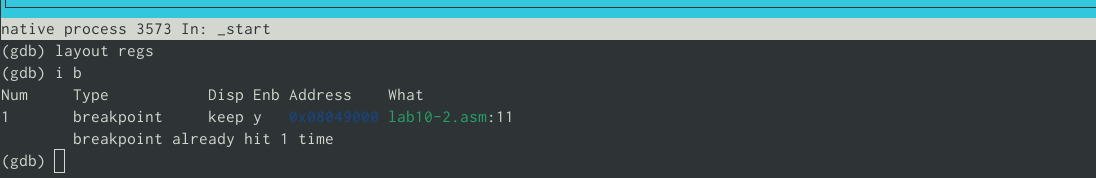


Рис. 9: -

добавим еще одну точку останова, но сделаем это по адресу (рис. 10)

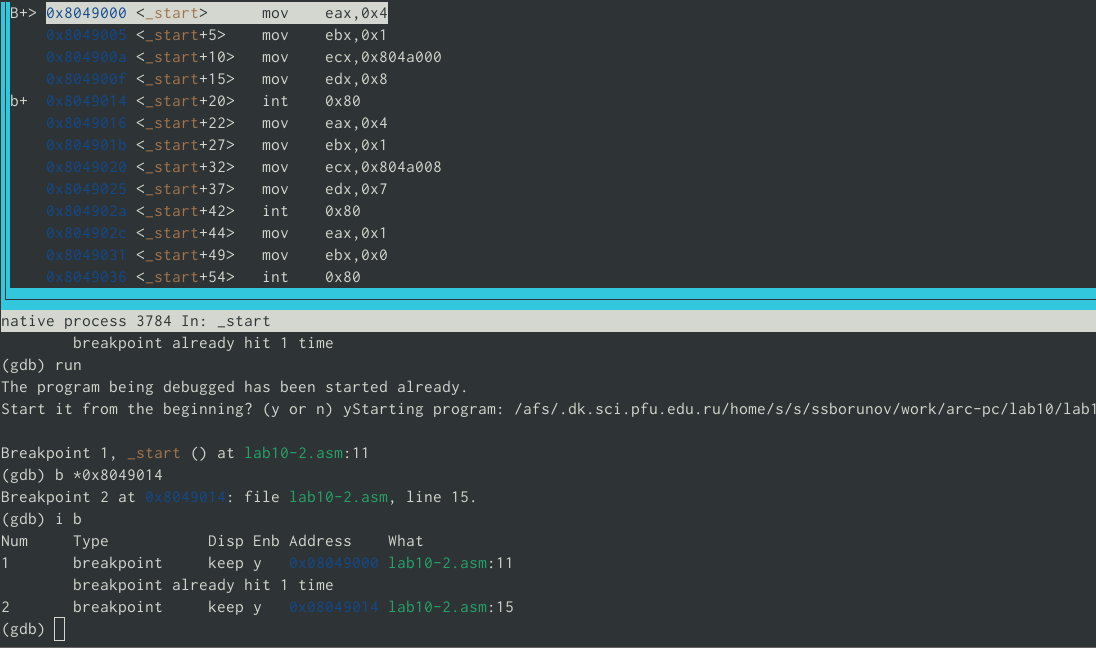


Рис. 10: -

Так же можно выводить значения регистров. Делается это командой i r. Псевдографика предствалена на (рис. 11)

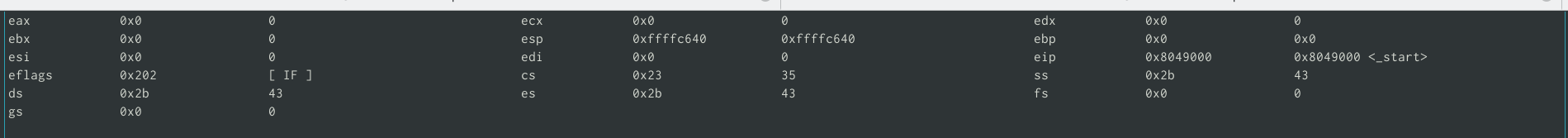


Рис. 11: -

В отладчике можно вывести текущее значение переменных. Сделать это можно например по имени (рис. 12) или по адресу (рис. 13)

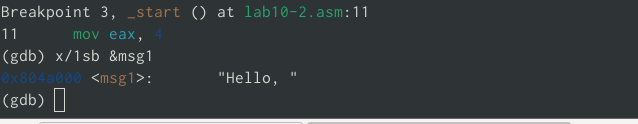


Рис. 12: -

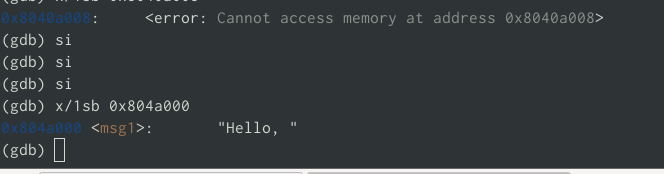


Рис. 13: -

Так же отладчик позволяет менять значения переменных прямо во время выполнения программы (рис. 14)

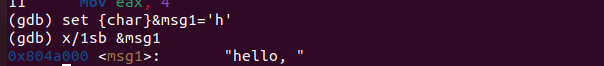


Рис. 14: -

Здесь тоже можно обращаться по адресам переменных(рис. 15). здесь был заменен первый символ переменной msg2 на символ отступа.

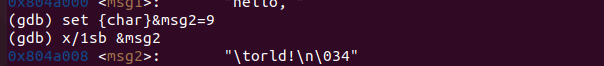


Рис. 15: -

Выоводить можно так же содержимое регисторов. Выведем значение edx в разных форматах: строчном, 16-ричном, двоичном(рис. 16)

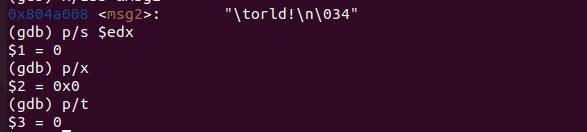


Рис. 16: -

Как и переменным, регистрам можно задавать значения.(рис. 17)

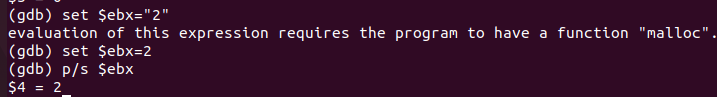


Рис. 17: -

Однако при попытке задать строчное значение, происходит ошибка.

Завершим работу в gdb командами continue, она закончит выполнение программы, и exit, она завершит сеанс gdb.

Скопируем файл из лабораторной 9, переименуем и создадим исполняемый файл. Откроем отладчик и зададим аргументы. Создадим точку останова на метке \_start и запустим программу(рис. 18)

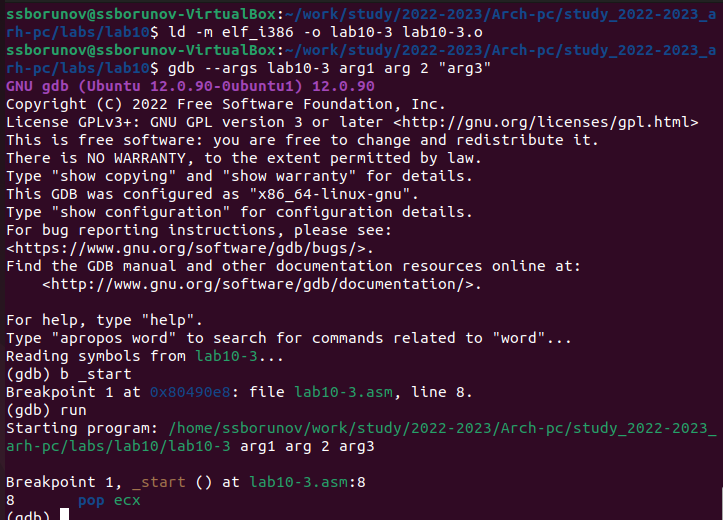


Рис. 18: -

Посмотрим на содержимое того,что расположено по адрессу, находящемуся в регистре esp (рис. 19)

Рис. 19: -

Рис. 19: -

Далее посмотрим на все остальные аргументы в стеке. Их адреса распологаются в 4 байтах друг от друга(именно столько заниемает элемент стека) (рис. 20)

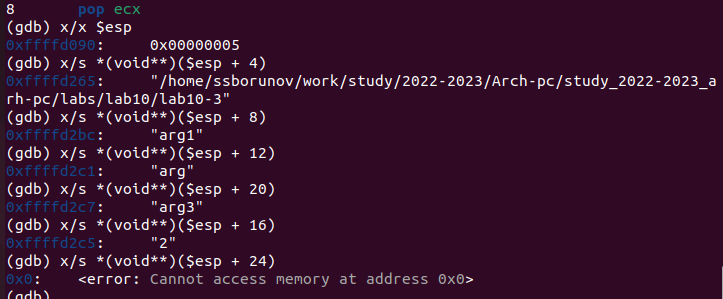


Рис. 20: -

# 3 Задания для самостоятельной работы

Программа из лабороторной 9, но с использованием подпрограмм (рис. 21)

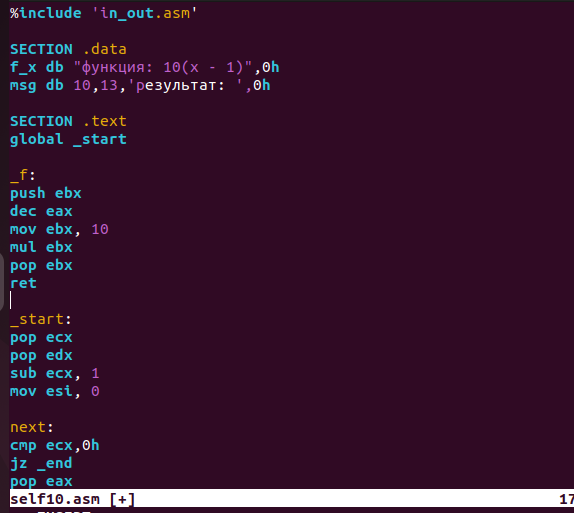


Рис. 21: -

и проверка ее работоспособности(рис. 22)

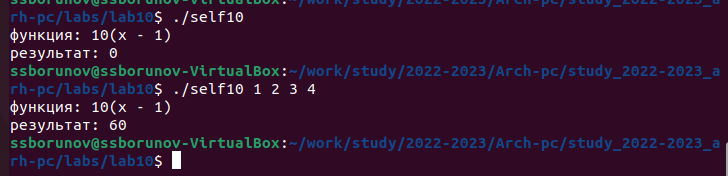


Рис. 22: -

Просмотр регистров, для поиска ошибки в программе из листинга 10.3 (рис. 23) и (рис. 24)

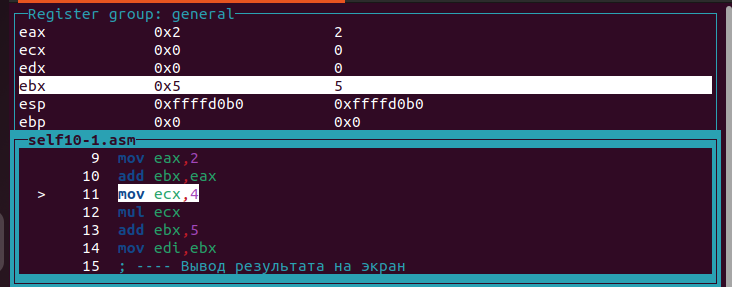


Рис. 23: -

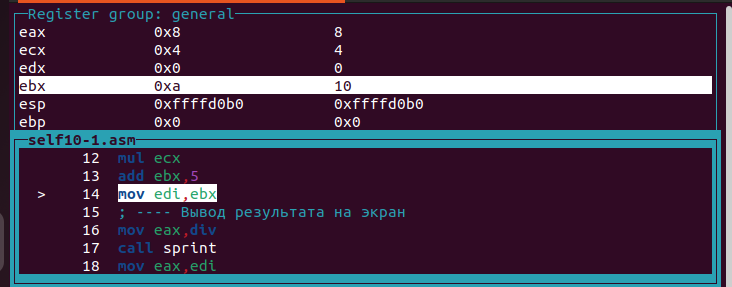


Рис. 24: -

Ошибка была в сторках

add ebx,eax  
mov ecx,4  
mul ecx  
add ebx,5  
mov edi,ebx

правильно работающая программа представлена на (рис. 25)

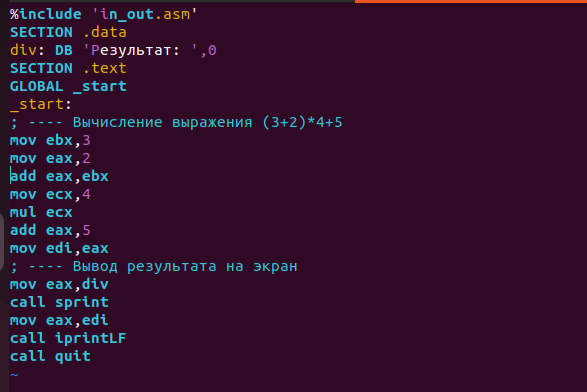


Рис. 25: -

Проверка корректронсти работы программы, после исправлений (рис. 26)

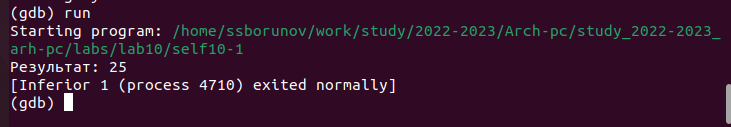


Рис. 26: -

# 4 Выводы

В результате выполнения работы, я научился организовывать код в подпрограммы и познакомился с базовыми функциями отладчика gdb.